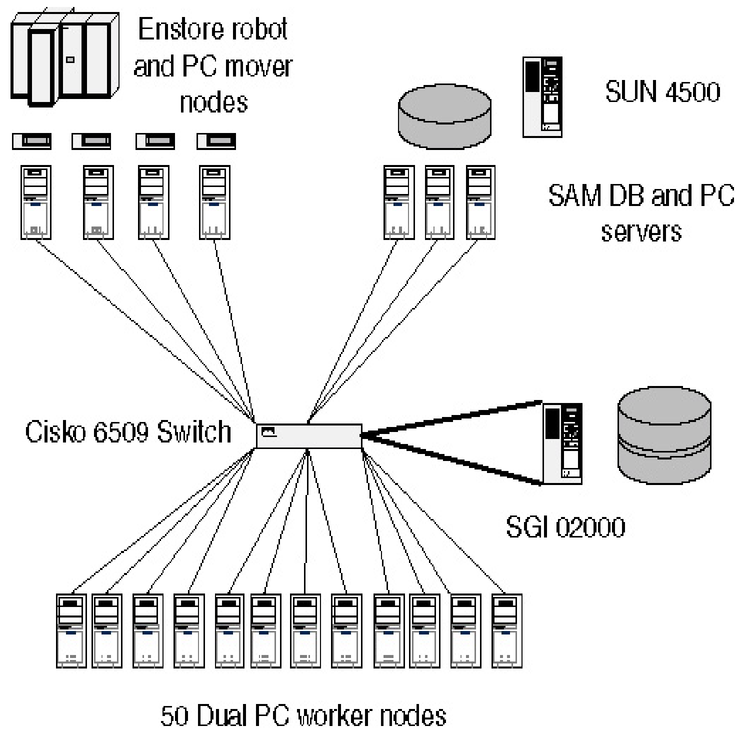
D0 farm status

Heidi Schellman
June 12, 00

D0 Farm needs

- 250K event size
- 50Hz trigger rate
  - peak rate of 12.5 MB/sec
  - DC is less but reprocessing will bring back up
- Reconstruction 5- 10 seconds/event
  on 500 MHz PIII
  - need 250-500 CPU's to handle peak rate
  - DC is 40% of peak
  - time constant for 1 GB file is 5- 10 hours.



I/O machine

- Purpose
  - split/merge of farm output
  - Serve home areas
  - Batch system control
  - File delivery master
- D0bbin
  - 4 CPU SGI 02000
  - 2 GB ethernet cards
  - 4 72 GB disk partitions (2 way stripe)
  - peak I/O rates of 40-60 MB/sec

Worker Nodes

- Dual Pentium III 500MHz
- 256MB/CPU
- 2 data disks (18 GB) + 6GB system
- Fast ethernet
- CD/floppy for system configuration

Plan to buy 50 new nodes this year
600 MHz 512 MB/CPU
Similar disk
Fast Ethernet
CD/floppy

Design Principles

- Use existing facilities
  - SAM/Enstore for data access and file tracking
  - Farm batch system (FBS) for most job control
- Keep D0 farm control scripts to a minimum
  - Batch system assigns machines
  - Data access system decides which file you get
- If worker process or machine dies, lose minimal number of files and don't affect other processes
- No heroic recovery measures, track and resubmit those files

Worker Configuration

- Workers act as generic FNAL farm machines
  - Only customization is pnfs for file delivery and home area mount
  - D0 environment downloads at job start
  - data access through SAM/encp/rcp, database server

- Batch system assigns workers to job, not D0FARM control process.
- D0FARM control never knows which workers are assigned to a job and does not need to.

- SAM processes currently run as part of worker batch job
  - Run them as local daemons with autorestart?
  - Run them as independent batch queues
    - This gives control over stop/start

Data Access is SAM/enstore

- Integrated data handling system
- File and process data base
- Data base server
- File servers
- Enstore File delivery systems
- Pnfs file system

Farm Perspective

Can tell it you want a set of files

Can ask for the 'next' file

Can flag file as processed or error

Can get detailed accounting on what happened

Farm accounts

- d0flib – library account has own ups/upd in /d0farm/fnal/ups – use this to install code
- d0fdev – special account for checks
- d0farm – account to run jobs from
  - Currently run jobs from prd3/farm_machinery/samtest.
- sam – sam account

- These are mounted on all machines, IO and workers

- I/O has 4 locally mounted stripe sets
  - /d0/stripeN …

- Each worker has local disks
  - /local/stage1/fbs_scratch 11 G for scratch
  - /local/stage2 ?? Unused
  - /local/d0 4G for constants downloads

Job submission

- Create project
  - Short csh script
  - Parameters are filename wildcard and reco version
  - Checks to see how many files of given description have been processed by reco version requested
  - Creates a project definition which is files with name x, tier digitized and no children processed through d0reco with version XXXX
- Create JDF file from template
  - Put in job parameters
  - Will change to python interface with FBS 3.0

- Submit job to farm and place info in log

Farm Batch System
Typical Farm Job

```
SECTION START
    EXEC=startjob
        parameters
    QUEUE=D0bbin
SECTION WORKER
    EXEC=runjob
        parameters
    NWORKERS=20
    QUEUE=D0worker
SECTION END
    EXEC=stopjob
        parameters
    QUEUE=D0bbin
    DEPEND
        WORKER(done)
```

- Queue tells the system what kind of machine to run on and how many.
- EXEC gives the script name and parameters
- DEPEND allows cleanup section to run when all worker sections are done.
- FBS assigns temporary disk on workers
- On end yanks disk and kills all processes.

Currently generated by shell script. Python API is now part of FBS 3.0 which is coming soon.

```
SECTION START_SAM
        EXEC=/home/d0farm/prd3/farm_machinery/samtest/start_sam_v6.csh
    preco03.07.00 protofarm prd3_single_preco03.07.00 new
    /home/d0farm/prd3/farm_machinery/samtest
    /home/d0farm/prd3/farm_machinery/samtest/Jun09 prd disk /d0/stripe2/samtest 50
        QUEUE=io_d0sgi
        NUMPROC=1
        MAILTO=schellma@d0mino.fnal.gov

    STDERR=/home/d0farm/prd3/farm_machinery/samtest/Jun09/prd3_single_preco03.07.00
    _%j_%n.err
    STDOUT=/home/d0farm/prd3/farm_machinery/samtest/Jun09/prd3_single_preco03.07.00
    _%j_%n.out
        NEED=1

SECTION WORKER_JOB
        EXEC=/home/d0farm/prd3/farm_machinery/samtest/d0reco_v6.sh preco03.07.00
    protofarm  prd3_single_preco03.07.00 /home/d0farm/prd3/farm_machinery/samtest
    /home/d0farm/prd3/farm_machinery/samtest/Jun09 prd disk
    d0farm@d0bbin:/d0/stripe2/samtest
        QUEUE=Worker_D0
        NUMPROC=11
        MAILTO=schellma@d0mino.fnal.gov

    STDERR=/home/d0farm/prd3/farm_machinery/samtest/Jun09/prd3_single_preco03.07.00
    _%j_%n.err
    STDOUT=/home/d0farm/prd3/farm_machinery/samtest/Jun09/prd3_single_preco03.07.00
    _%j_%n.out
        NEED=1
        DEPEND=started(START_SAM)

SECTION END
        EXEC=/home/d0farm/prd3/farm_machinery/samtest/stop_sam_v6b.csh
    preco03.07.00 protofarm prd3_single_preco03.07.00
    /home/d0farm/prd3/farm_machinery/samtest
    /home/d0farm/prd3/farm_machinery/samtest/Jun09 prd disk /d0/stripe2/samtest
        QUEUE=io_d0sgi
        NUMPROC=1
        MAILTO=schellma@d0mino.fnal.gov

    STDERR=/home/d0farm/prd3/farm_machinery/samtest/Jun09/prd3_single_preco03.07.00
    _%j_%n.err

    STDOUT=/home/d0farm/prd3/farm_machinery/samtest/Jun09/prd3_single_preco03.07.00
    _%j_%n.out
        NEED=1
        DEPEND=ended(WORKER_JOB)
```

Job parameters

- Input as parameters when jdf created
  - Reco_vers
  - Project definition name
  - Sam station name
  - Command directory
  - Lsf output directory (must be cross mounted)
  - IO machine spool disk
  - IO machine log directory
  - Sam db version
  - Optional tag for interactive jobs
- Generated by batch system
  - LSF job id
  - Worker node
  - Batch process number
  - Local scratch area
- Passed between sections
  - Consumer ID file
- Derived
  - Analysis project name (from lsf)
  - Subsidiary disk areas

Start Section

- Set up products and output directories on d0bbin
- Start the sam project
- Start a sam consumer
- Store consumer ID in special file tagged by lsf jobid on shared disk.
- Create output directories on I/O machine
- Go into wait state until get end signal (currently deletion of the CID file.
- Parameters:

Worker Section

- Generate analysis name from job id
- Get CID number from CID disk file tagged by jobid
- Wait N* jobnumber seconds
- Check that project is in fact running
- Download D0 environment
- Start SAM stager (should be made independent)
- Ask for next file
- Process file
- Generate metadata for output file
- Store output file and metadata on output buffer
- Store output logs on output buffer
- Inform SAM of success
- Ask for next file
- On error or end of list, terminate.

End Section

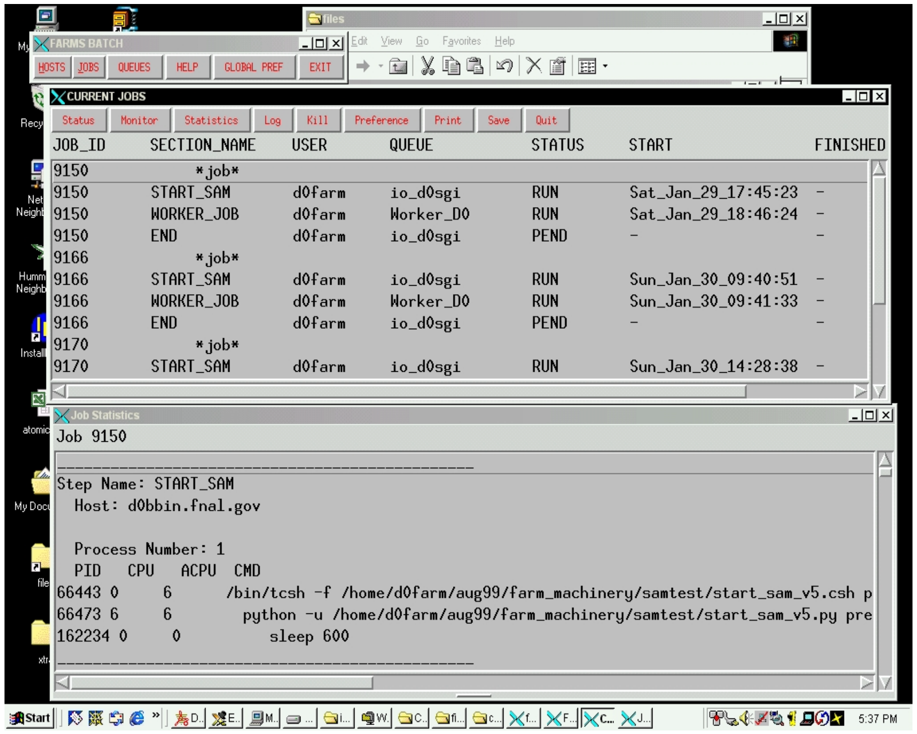
- Create job summary
- Stop the sam project
- Send message to Start process telling it to shut down
  - Done by moving the CID file
- (Optional) Start file merge/store of output files.
- Copy log files on to I/O node spool disks from shared directories

Storing files

- Currently do as independent step on I/O node
  - storeallfiles.py <fullpath> stores all files with metadata in a given directory back into sam
  - It has lots of loopbacks
- This mimics what will be done for merging on I/O node
- Not very robust at all.

Diagnostics

- Farm batch system
  - Farms hosts tells what is running where
  - Farms status line mode list of processes
  - Farms monitor – gui
- Sam system
  - SQL queries
  - SAM Data Browsers
    - Datafiles
    - Project definitions
    - Analysis projects
    - Running projects
- Check_project scripts
  - Issues command line SQL with parentage information



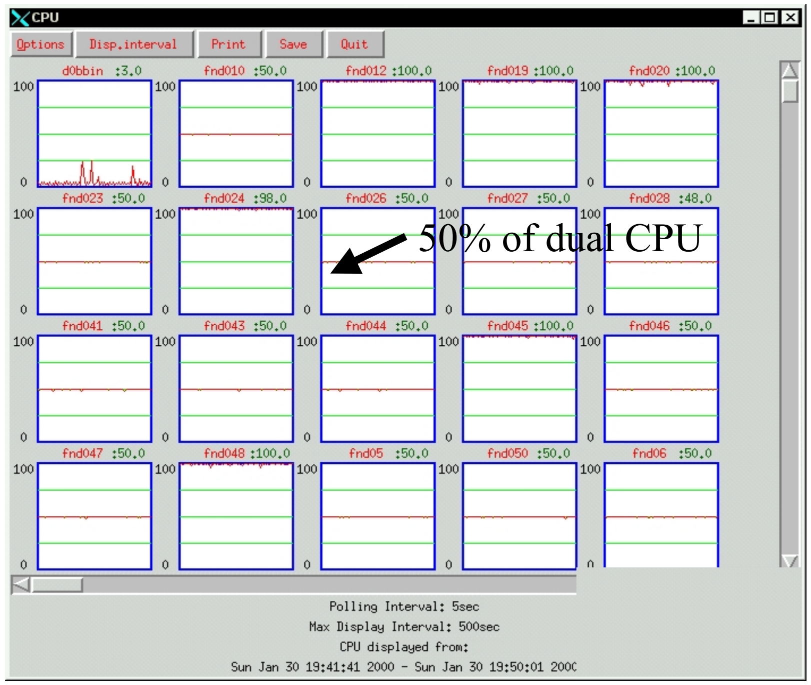
```
JOB_ID   SECTION_NAME   USER     QUEUE        STATUS   START                  FINISHED
9150           *job*
9150     START_SAM      d0farm   io_d0sgi     RUN      Sat_Jan_29_17:45:23    -
9150     WORKER_JOB     d0farm   Worker_D0    RUN      Sat_Jan_29_18:46:24    -
9150     END            d0farm   io_d0sgi     PEND     -                      -
9166           *job*
9166     START_SAM      d0farm   io_d0sgi     RUN      Sun_Jan_30_09:40:51    -
9166     WORKER_JOB     d0farm   Worker_D0    RUN      Sun_Jan_30_09:41:33    -
9166     END            d0farm   io_d0sgi     PEND     -                      -
9170           *job*
9170     START_SAM      d0farm   io_d0sgi     RUN      Sun_Jan_30_14:28:38    -
```

```
Job 9150
------------------------------------------------
Step Name: START_SAM
  Host: d0bbin.fnal.gov

  Process Number: 1
  PID    CPU    ACPU   CMD
66443 0       6       /bin/tcsh -f /home/d0farm/aug99/farm_machinery/samtest/start_sam_v5.csh p
66473 6       6         python -u /home/d0farm/aug99/farm_machinery/samtest/start_sam_v5.py pre
162234 0      0            sleep 600
------------------------------------------------
```

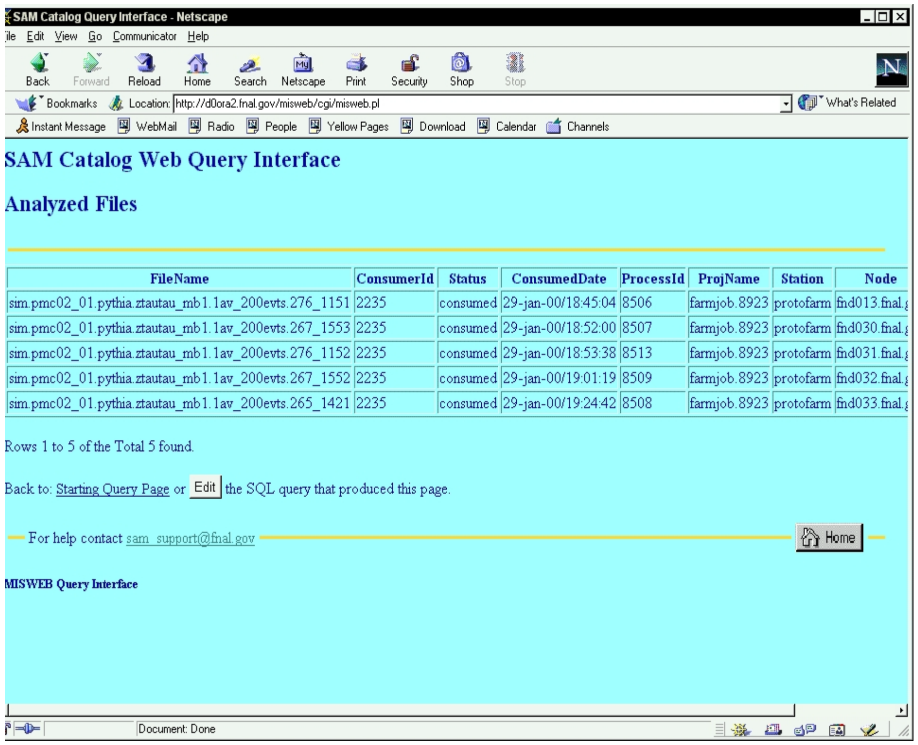
Farm Batch System Monitor

100% of dual



```
Polling Interval: 5sec
Max Display Interval: 500sec
CPU displayed from:
Sun Jan 30 19:41:41 2000 - Sun Jan 30 19:50:01 2000
```

Jobs use 100% of CPU


SAM Catalog Query Interface - Netscape
ile Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Shop Stop
Bookmarks Location: http://d0ora2.fnal.gov/misweb/cgi/misweb.pl
What's Related
Instant Message WebMail Radio People Yellow Pages Download Calendar Channels
Document: Done

SAM Catalog Web Query Interface

Analyzed Files

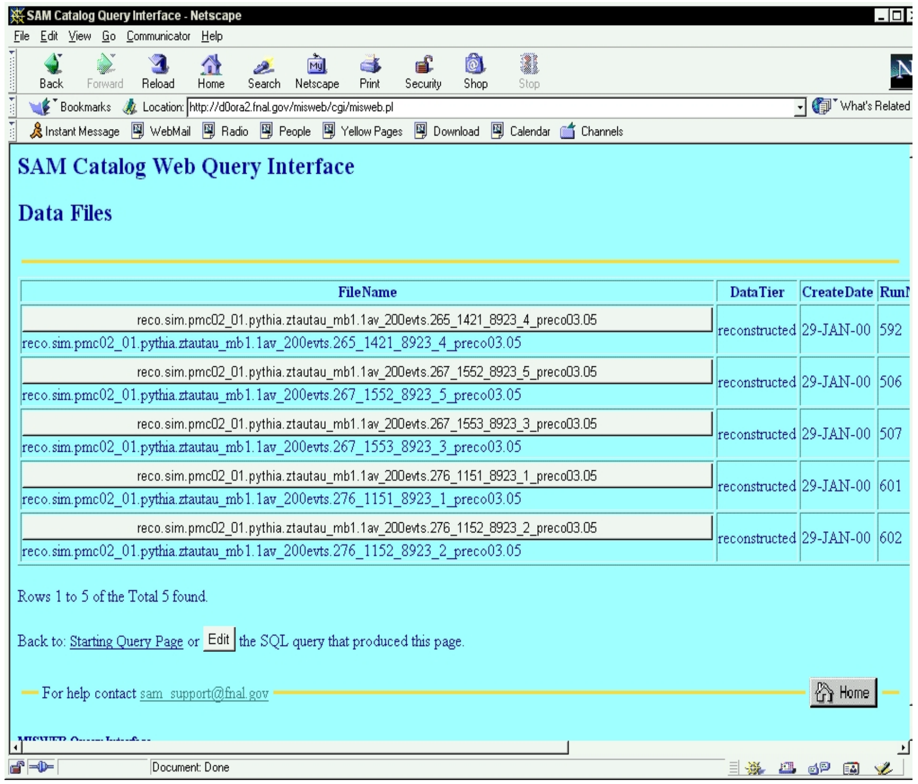
FileName	ConsumerId	Status	ConsumedDate	ProcessId	ProjName	Station	Node
sim.pmc02_01.pythia.ztautau_mb1.1av_200evts.276_1151	2235	consumed	29-jan-00/18:45:04	8506	farmjob.8923	protofarm	fnd013.fnal.
sim.pmc02_01.pythia.ztautau_mb1.1av_200evts.267_1553	2235	consumed	29-jan-00/18:52:00	8507	farmjob.8923	protofarm	fnd030.fnal.
sim.pmc02_01.pythia.ztautau_mb1.1av_200evts.276_1152	2235	consumed	29-jan-00/18:53:38	8513	farmjob.8923	protofarm	fnd031.fnal.
sim.pmc02_01.pythia.ztautau_mb1.1av_200evts.267_1552	2235	consumed	29-jan-00/19:01:19	8509	farmjob.8923	protofarm	fnd032.fnal.
sim.pmc02_01.pythia.ztautau_mb1.1av_200evts.265_1421	2235	consumed	29-jan-00/19:24:42	8508	farmjob.8923	protofarm	fnd033.fnal.

Rows 1 to 5 of the Total 5 found.

Back to: Starting Query Page or Edit the SQL query that produced this page.

For help contact sam_support@fnal.gov Home

MISWEB Query Interface


SAM Catalog Query Interface - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Shop Stop
Bookmarks Location: http://d0ora2.fnal.gov/misweb/cgi/misweb.pl
What's Related
Instant Message WebMail Radio People Yellow Pages Download Calendar Channels
Document: Done

SAM Catalog Web Query Interface

Data Files

FileName	DataTier	CreateDate	Run
reco.sim.pmc02_01.pythia.ztautau_mb1.1av_200evts.265_1421_8923_4_preco03.05 reco.sim.pmc02_01.pythia.ztautau_mb1.1av_200evts.265_1421_8923_4_preco03.05	reconstructed	29-JAN-00	592
reco.sim.pmc02_01.pythia.ztautau_mb1.1av_200evts.267_1552_8923_5_preco03.05 reco.sim.pmc02_01.pythia.ztautau_mb1.1av_200evts.267_1552_8923_5_preco03.05	reconstructed	29-JAN-00	506
reco.sim.pmc02_01.pythia.ztautau_mb1.1av_200evts.267_1553_8923_3_preco03.05 reco.sim.pmc02_01.pythia.ztautau_mb1.1av_200evts.267_1553_8923_3_preco03.05	reconstructed	29-JAN-00	507
reco.sim.pmc02_01.pythia.ztautau_mb1.1av_200evts.276_1151_8923_1_preco03.05 reco.sim.pmc02_01.pythia.ztautau_mb1.1av_200evts.276_1151_8923_1_preco03.05	reconstructed	29-JAN-00	601
reco.sim.pmc02_01.pythia.ztautau_mb1.1av_200evts.276_1152_8923_2_preco03.05 reco.sim.pmc02_01.pythia.ztautau_mb1.1av_200evts.276_1152_8923_2_preco03.05	reconstructed	29-JAN-00	602

Rows 1 to 5 of the Total 5 found.

Back to: Starting Query Page or Edit the SQL query that produced this page.

For help contact sam_support@fnal.gov Home

Query to see which input files were processed by a job

Check to see if output files were stored properly

Results of typical farm test

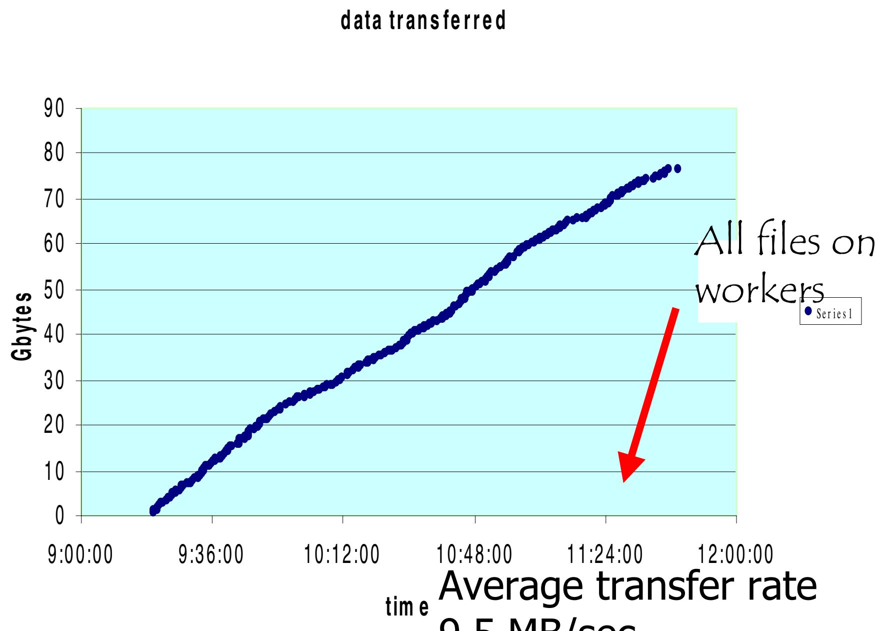
- Create 4 jobs with 25-180 files in each (350 total)
- Submit 4 jobs to the farms using 10-30 workers each (occupy 95/100)
- Process those files through official reconstruction executable
- Files are 200-700 MB Monte Carlo, take 2-10 hours to process.
- 14 tapes read by 5 tape drives (3MB/sec max/drive)
- Output written to I/O node for later dump to tape
- This is almost* equivalent to starting a production 100 processor farm from a cold start.

*exception is tape drive speed -> 12MB/sec, did not do output to tape

Data transfer to workers

Fire up 4 jobs
Zee, zmumu, ttbar
Qcdpt>80

322 files
95 worker CPU
5 tape drives
14 different tapes



Average transfer rate
9.5 MB/sec
Peak ~ 15 MB/sec

Things to do

- Cleanup
  - Better python interface to sam
  - User python API for FBS when it arrives
  - Split stagers out of worker jobs
  - Rewrite scripts to use components
- Job control and submission
  - Create ‘Job’ object rather than command line
  - Make ‘Job’ a subset of project instead of other way around? Use SAM resubmit capability
  - Automate job submission
- Split/Merge
  - Get sam metadata for split merge
  - Get copyevpack going to merge
  - Get merge algorithm to choose files
  - Improve file storage
- Diagnostics/Control
  - Show all running projects
  - Show all running jobs
  - Ability to kill individual d0reco processes

How D0reco is currently built

- Log onto d0lxbld4
- Go to scratch area
- setenv PATH /d0dist/dist/release/t00.92.00/d0reco/scripts:$PATH
- buildfarmreco test

- Makes file t00.92.00-test.tar

- ftp to the ~d0farm/d0reco area on one of the worker nodes
- ~d0farm/untarme t00.92.00-test.tar

- mv t00.92.00-test t00.92.00
- tarreco t00.92.00
  - This makes a t00.92.00.tar in the t00.92.00 directory
  - Farm asks for ~/d0reco/t00.92.00/t00.92.00.tar right now.

- (some of these steps could be streamlined,this was designed so you could test before running)

How a job is submitted

- makeandrun <filename fragment> <diskid>

- from the ~/prd3/farm_machinery/samtest area

- Makeandrun has preco03.07.00 hardwired for now.

- Creates project
- Creates JDF file
- Submits job

- check_project <filename fragment> <recoversion>